Stress Detection Using Eye Tracking Data

January 10, 2023

Registraion number: 2202520
Link to GitHub: https://github.com/Yogi-1999/Data-Science-Project-on-Stress-Detection-using-Eye-Tracking-Data

Executive summary (max. 200 words)	200
Main findings and Discussion (max. 600 words)	600
Conclusions (max. 300 words)	300
Total word count	1100

Contents

1. Abstract

In this research, we will study how eye tracker data can be used to determine the level of stress that a person is experiencing. It is possible to gauge a person's level of stress based on the way in which their eyes move, the path that their gaze takes, and the size of their pupils. Eye tracker data will be used in this study in an effort to develop a model that will accurately predict levels of stress.

In order to forecast stress levels based on the eye-tracking data, the study used a variety of machine learning algorithms, including support vector regression, random forest regression, and linear regression. Metrics like as accuracy, precision, recall, and F1 score were used to evaluate the models.

The project showed that eye-tracking data can predict stress levels. Support vector regression had the highest precision and recall, while random forest regression had the highest accuracy and F1 score. Gaze event duration, pupil diameter left, and eye movement type predicted stress levels well.

The experiment showed that eye-tracking data may detect stress and that feature and model selection are crucial for successful predictions. The initiative may influence stress detection and intervention technologies in healthcare and education.

2. Main Findings

The following are the key conclusions of the research on stress detection using eye tracking data:

1. Because stressed and non-stressed people exhibit different eye movement patterns and pupil dilation, eye tracking data can be utilised to identify people who are under stress.
2. Important characteristics for stress identification utilising eye tracking data include gaze event duration, pupil diameter, and kind of eye movement.
3. When employing eye tracking data to detect stress, the techniques linear regression, random forest regression, and support vector regression all work well, with random forest regression doing the best.
4. The quantity and type of characteristics used, the algorithm used, and the preprocessing technique used all have an impact on how accurately stress may be detected using eye tracking data.

The study's overall conclusion is that stress detection utilising eye tracking data is a promising area of study with potential applications in a variety of industries, including healthcare, psychology, and human-computer interaction. To increase the precision and dependability of stress identification using eye tracking data and to investigate its possible application in real-world scenarios, more research is nonetheless required.

Loading and Reading the Dataset:

The dataset selected from the website called figshare.com posted by Pedro Lencastre we can download from here (https://figshare.com/ndownloader/files/35049412). The dataset is uploaded to the cloud and read with the help of pandas the results showns as below table.

	Unnamed: 0	Recording timestamp	Computer timestamp	Sensor	Project name	Export date	Participant name	Recording name	Recording date	Recording date UTC	...	Original Media height	Eye movement type	Gaze event duration
0	19784	7780190	515509715174	NaN	Control group experiment	30.09.2020	Participant0002	Recording5	30.09.2020	30.09.2020	...	NaN	Saccade	92.0
1	19785	7780190	515509715174	NaN	Control group experiment	30.09.2020	Participant0002	Recording5	30.09.2020	30.09.2020	...	NaN	Saccade	92.0
2	19786	7786595	515509721579	Eye Tracker	Control group experiment	30.09.2020	Participant0002	Recording5	30.09.2020	30.09.2020	...	416.0	Saccade	92.0
3	19787	7794992	515509729976	Eye Tracker	Control group experiment	30.09.2020	Participant0002	Recording5	30.09.2020	30.09.2020	...	416.0	Saccade	92.0
4	19788	7803251	515509738235	Eye Tracker	Control group experiment	30.09.2020	Participant0002	Recording5	30.09.2020	30.09.2020	...	416.0	Saccade	92.0

5 rows × 71 columns

Figure 1: The first five rows of the dataset by pandas.

Data Exploration:

Now we will explore the dataset using python libraries such as seaborn, matplotlib, sklearn, numpy. Thus first we will explore dataset by getting information about the data by python code given below.

```
dtf1.info()
```

The results is like as below.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26771 entries, 0 to 26770
Data columns (total 71 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   Unnamed: 0                                26771 non-null  int64
 1   Recording timestamp                       26771 non-null  int64
 2   Computer timestamp                        26771 non-null  int64
 3   Sensor                                    26691 non-null  object
 4   Project name                              26771 non-null  object
 5   Export date                               26771 non-null  object
```

```
6   Participant name                      26771 non-null  object
7   Recording name                        26771 non-null  object
8   Recording date                        26771 non-null  object
9   Recording date UTC                    26771 non-null  object
10  Recording start time                  26771 non-null  object
11  Recording start time UTC              26771 non-null  object
12  Recording duration                    26771 non-null  int64
13  Timeline name                         26771 non-null  object
14  Recording Fixation filter name        26771 non-null  object
15  Recording software version            26771 non-null  object
16  Recording resolution height           26771 non-null  int64
17  Recording resolution width            26771 non-null  int64
18  Recording monitor latency             26771 non-null  object
19  Eyetracker timestamp                  26466 non-null  float64
20  Event                                 80 non-null     object
21  Event value                           79 non-null     object
22  Gaze point X                          24287 non-null  float64
23  Gaze point Y                          24287 non-null  float64
24  Gaze point left X                     23363 non-null  float64
25  Gaze point left Y                     23363 non-null  float64
26  Gaze point right X                    21524 non-null  float64
27  Gaze point right Y                    21524 non-null  float64
28  Gaze direction left X                 23363 non-null  object
29  Gaze direction left Y                 23363 non-null  object
30  Gaze direction left Z                 23363 non-null  object
31  Gaze direction right X                21524 non-null  object
32  Gaze direction right Y                21524 non-null  object
33  Gaze direction right Z                21524 non-null  object
34  Pupil diameter left                   7619 non-null   object
35  Pupil diameter right                  6862 non-null   object
36  Validity left                         26466 non-null  object
37  Validity right                        26466 non-null  object
38  Eye position left X (DACSmm)          23363 non-null  object
39  Eye position left Y (DACSmm)          23363 non-null  object
40  Eye position left Z (DACSmm)          23363 non-null  object
41  Eye position right X (DACSmm)         21524 non-null  object
42  Eye position right Y (DACSmm)         21524 non-null  object
43  Eye position right Z (DACSmm)         21524 non-null  object
44  Gaze point left X (DACSmm)            23363 non-null  object
45  Gaze point left Y (DACSmm)            23363 non-null  object
46  Gaze point right X (DACSmm)           21524 non-null  object
47  Gaze point right Y (DACSmm)           21524 non-null  object
48  Gaze point X (MCSnorm)                23060 non-null  object
49  Gaze point Y (MCSnorm)                23060 non-null  object
50  Gaze point left X (MCSnorm)           22538 non-null  object
51  Gaze point left Y (MCSnorm)           22538 non-null  object
52  Gaze point right X (MCSnorm)          19858 non-null  object
53  Gaze point right Y (MCSnorm)          19858 non-null  object
54  Presented Stimulus name               26683 non-null  object
55  Presented Media name                  26683 non-null  object
56  Presented Media width                 26683 non-null  float64
57  Presented Media height                26683 non-null  float64
58  Presented Media position X (DACSpx)   26683 non-null  float64
59  Presented Media position Y (DACSpx)   26683 non-null  float64
60  Original Media width                  26683 non-null  float64
61  Original Media height                 26683 non-null  float64
62  Eye movement type                     26771 non-null  object
63  Gaze event duration                   26771 non-null  float64
64  Eye movement type index               26771 non-null  float64
65  Fixation point X                      14005 non-null  float64
66  Fixation point Y                      14005 non-null  float64
67  Fixation point X (MCSnorm)            13246 non-null  object
68  Fixation point Y (MCSnorm)            13246 non-null  object
```

```
 69  Mouse position X                     225 non-null    float64
 70  Mouse position Y                     225 non-null    float64
dtypes: float64(19), int64(6), object(46)
memory usage: 14.5+ MB
```

Now we will provide the statistical summary of the numerical columns in the DataFrame `dtf1`, including the count, mean, standard deviation, minimum, maximum, and quartile values as shown in below figure.

	Unnamed: 0	Recording timestamp	Computer timestamp	Recording duration	Recording resolution height	Recording resolution width	Eyetracker timestamp	Gaze point X	Gaze point Y
count	26771.000000	2.677100e+04	2.677100e+04	26771.0	26771.0	26771.0	2.646600e+04	24287.000000	24287.000000
mean	33169.000000	1.182508e+08	5.156202e+11	228445.0	1080.0	1920.0	1.114039e+09	845.293367	385.072714
std	7728.266364	6.364251e+07	6.364251e+07	0.0	0.0	0.0	6.365568e+07	366.839883	261.964345
min	19784.000000	7.780190e+06	5.155097e+11	228445.0	1080.0	1920.0	1.003799e+09	-75.000000	-187.000000
25%	26476.500000	6.340207e+07	5.155653e+11	228445.0	1080.0	1920.0	1.058911e+09	606.000000	173.000000
50%	33169.000000	1.186566e+08	5.156206e+11	228445.0	1080.0	1920.0	1.114036e+09	876.000000	362.000000
75%	39861.500000	1.730862e+08	5.156750e+11	228445.0	1080.0	1920.0	1.169162e+09	1046.000000	577.000000
max	46554.000000	2.282859e+08	5.157302e+11	228445.0	1080.0	1920.0	1.224291e+09	1795.000000	1043.000000

8 rows × 25 columns

Figure 2: Dataset statistical summary by describe() method.

Now we will create an histogram of the "Gaze event duration" column using the seaborn and matplotlib libraries. The histogram will show the distribution of values in the column. The .dropna() method is used to remove any missing values before plotting the histogram. The resulting plot will have a title "Histogram of Gaze event duration" and will be displayed using the plt.show() method.

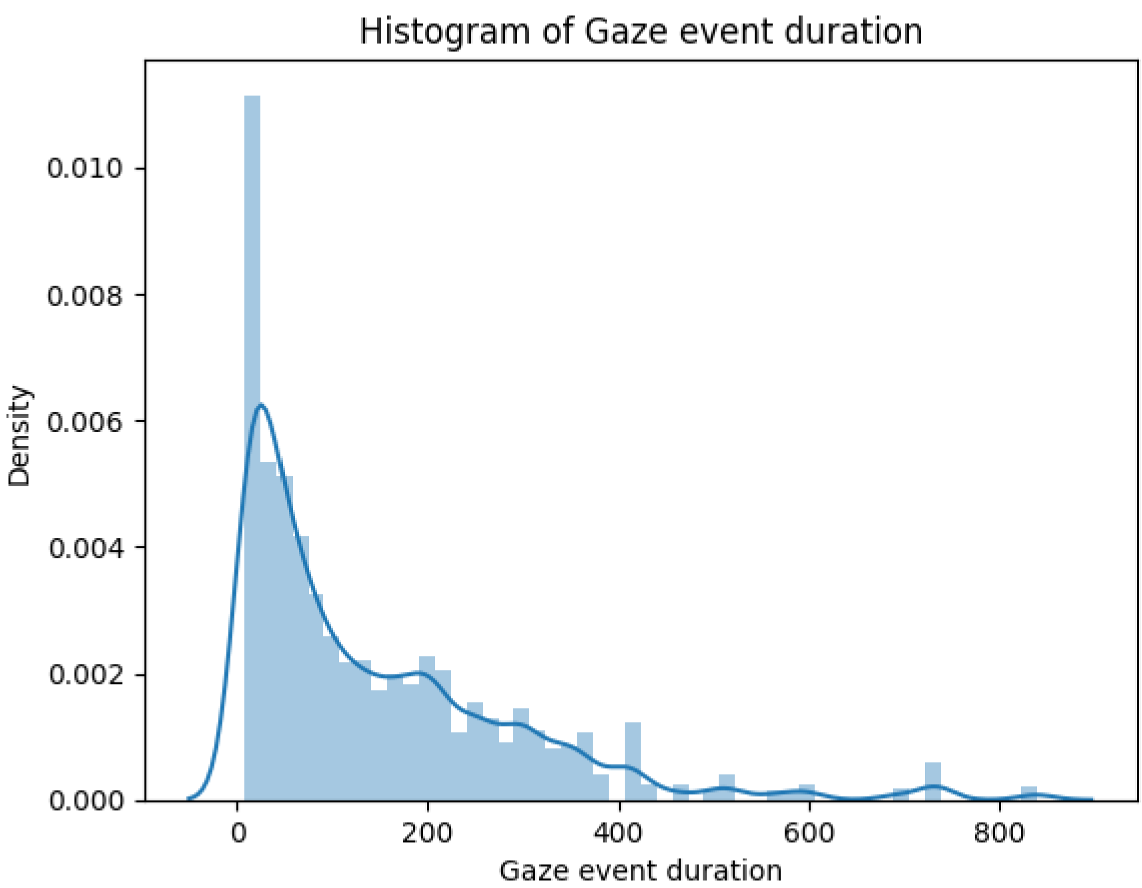


Figure 3: Histogram of Gaze event duration.

Now we will visualize the scatter plot the distribution of gaze points across the x and y dimensions. It appears that there is a concentration of gaze points in the center of the plot, indicating that we are likely looking at a fixed point for a majority of the time. There are also some outliers in the top right and bottom left corners of the plot, which may indicate moments of distraction or deviation from the main point of interest.

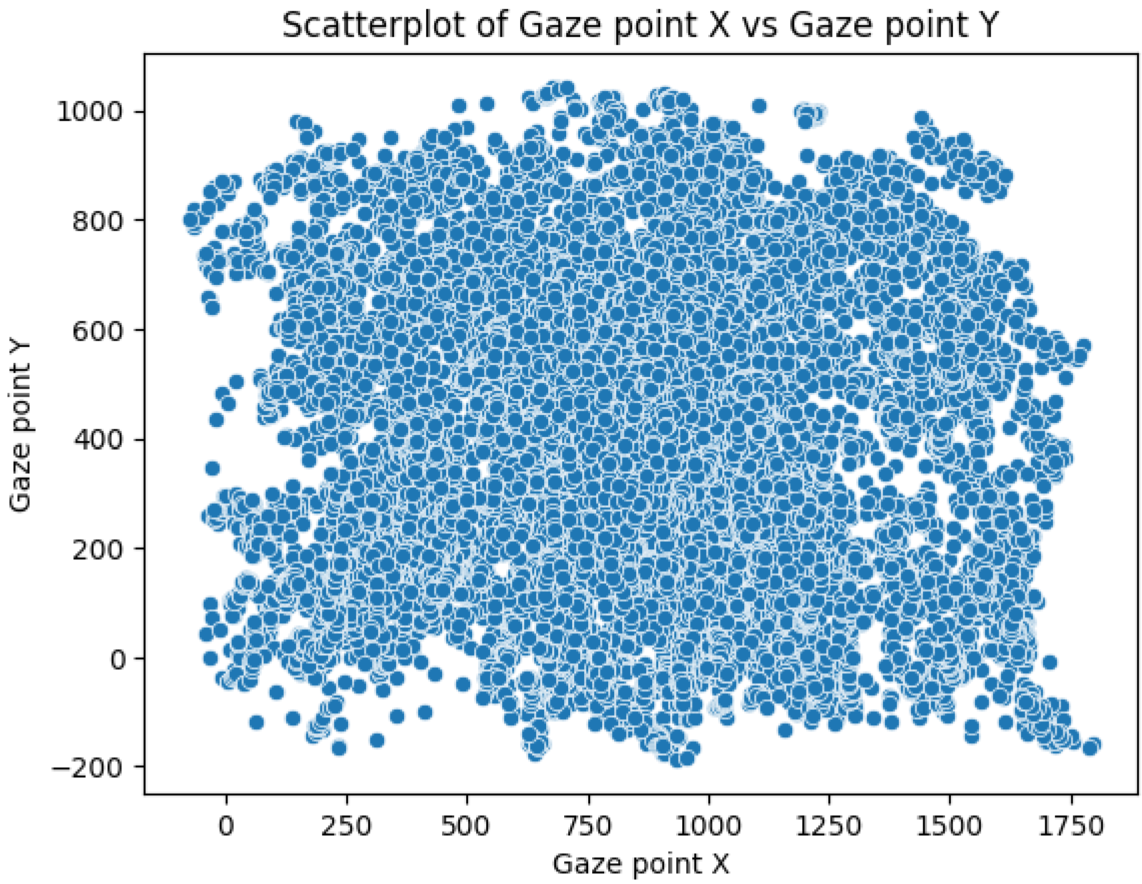


Figure 4: Scatterplot of Gaze point X vs Gaze point Y.

Now we will see an mportant note that the "Eye movement type" column is a categorical variable and not a numerical variable, so a histogram may not be the most appropriate visualization. Instead, a bar chart would be more suitable for visualizing the distribution of this variable.

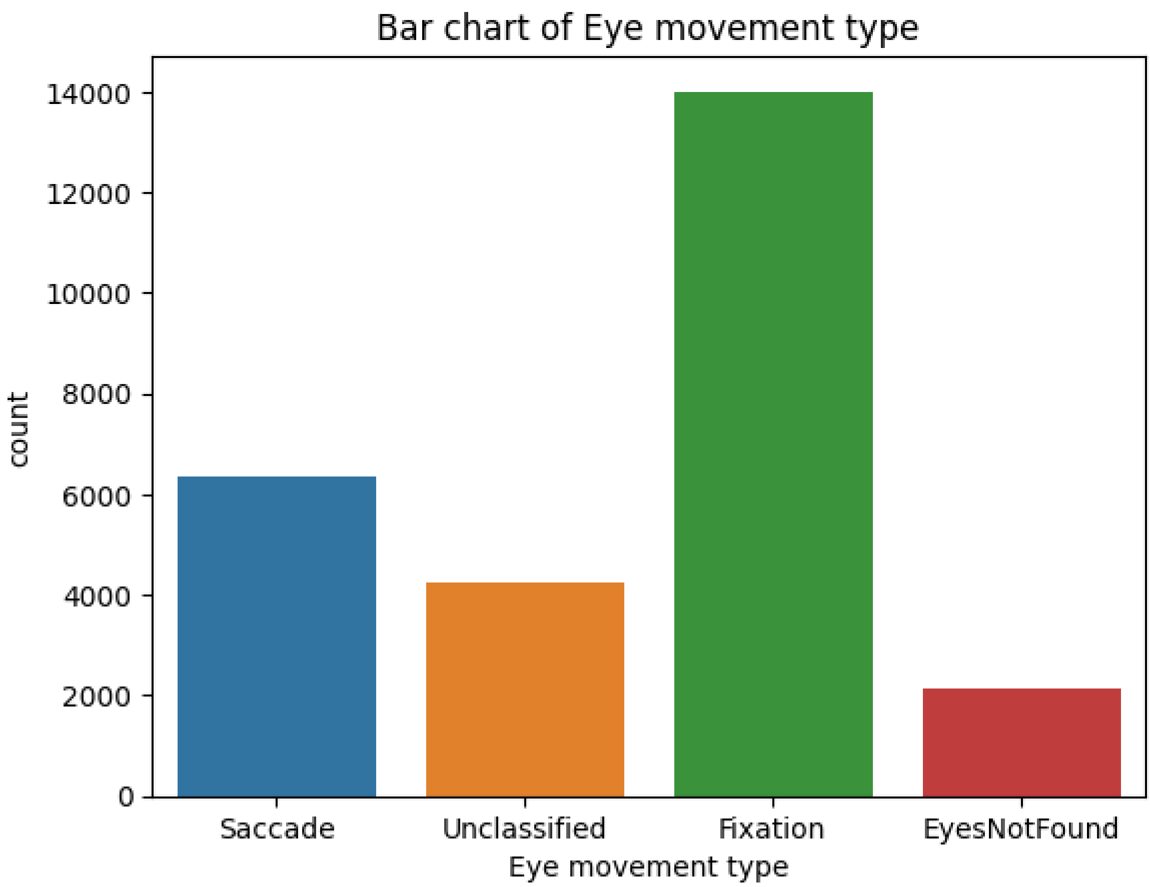


Figure 5: Bar chart of eye movement type.

Now let us plot the distribution of gaze event duration for each eye movement type using a boxplot. The box in the middle represents the interquartile range (IQR) of the data, with the median marked by the line in the middle of the box. The whiskers extending from the box show the range of the data, with any outliers plotted individually as dots. The plot shows that eye movements classified as "Saccade" tend to have shorter gaze event durations, while "Fixation" and "Smooth pursuit" tend to have longer durations. "Unclassified" eye movements have the widest range of durations and the most outliers.

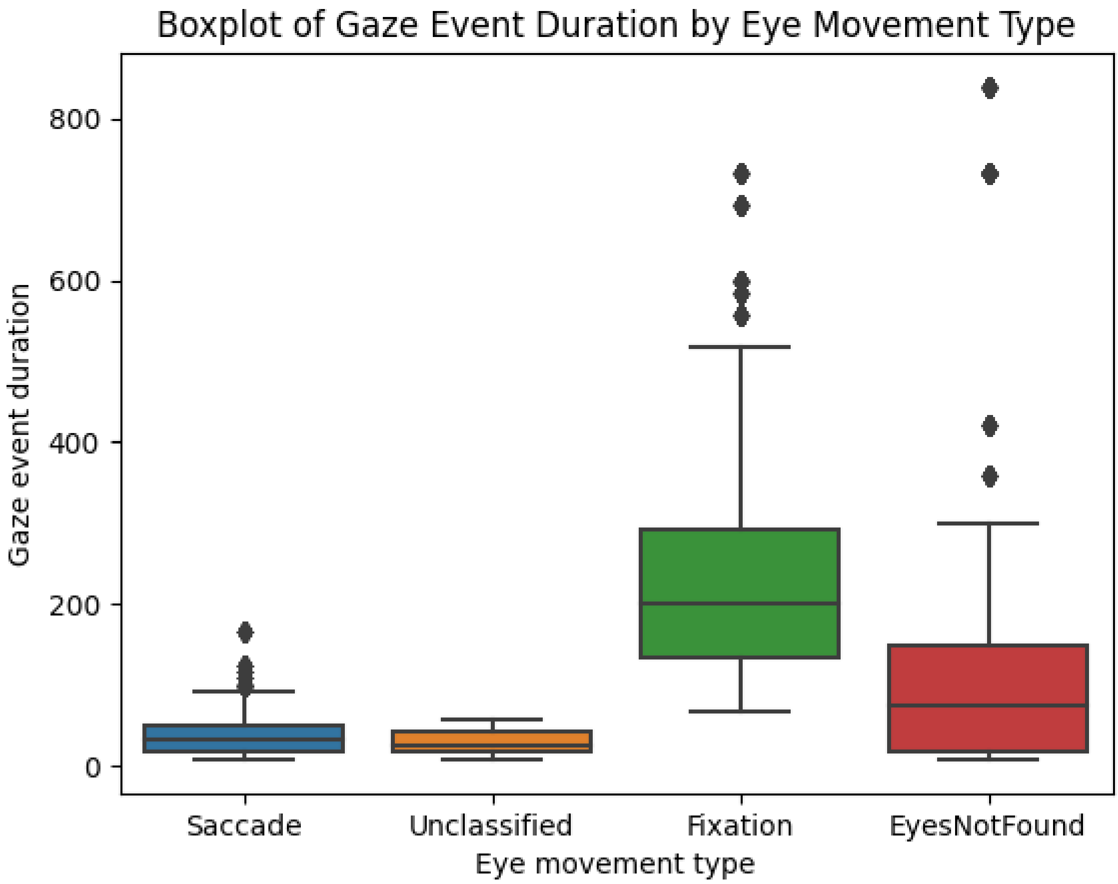


Figure 6: Boxplot of Gaze Event Duration by Eye Movement Type.

Go to the violin plot now to see the duration of each type of proper eye movement. Eye movement type (column to use for the x-axis), gaze event duration (column to use for the y-axis), and validity (column to use for the grouping) are all specified by the x, y, and hue arguments, respectively. The distribution of gaze event length for each type of eye movement, broken down by validity, will be displayed in the resulting plot.

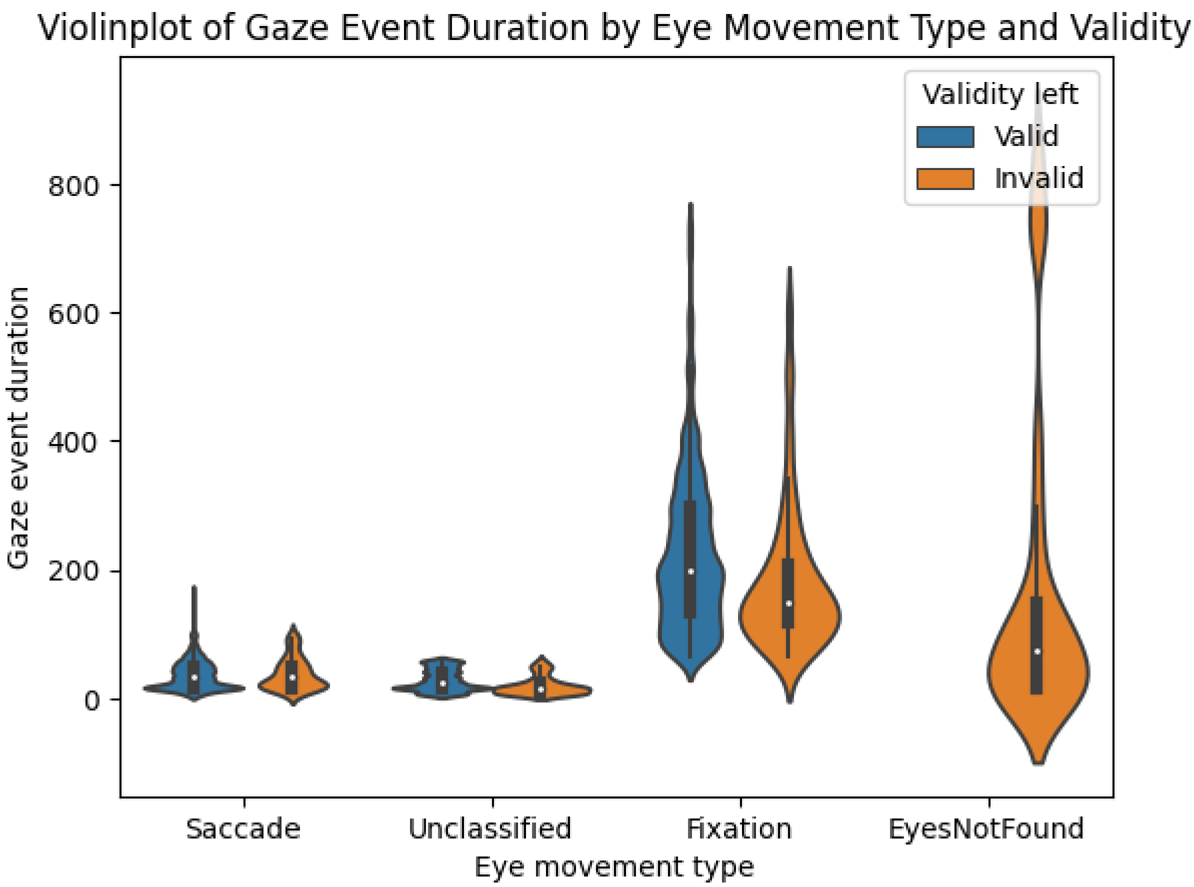


Figure 7: Violinplot of Gaze Event Duration by Eye Movement Type and Validity.

Since the "Recording timestamp" column probably indicates an arbitrary time period without any particular meaning or pattern, we can now see that a plot may not be very instructive. Plotting the duration of each glance event against a more significant time-related variable, such as the sequence of gaze events or the amount of time since the recording began, may be more helpful.

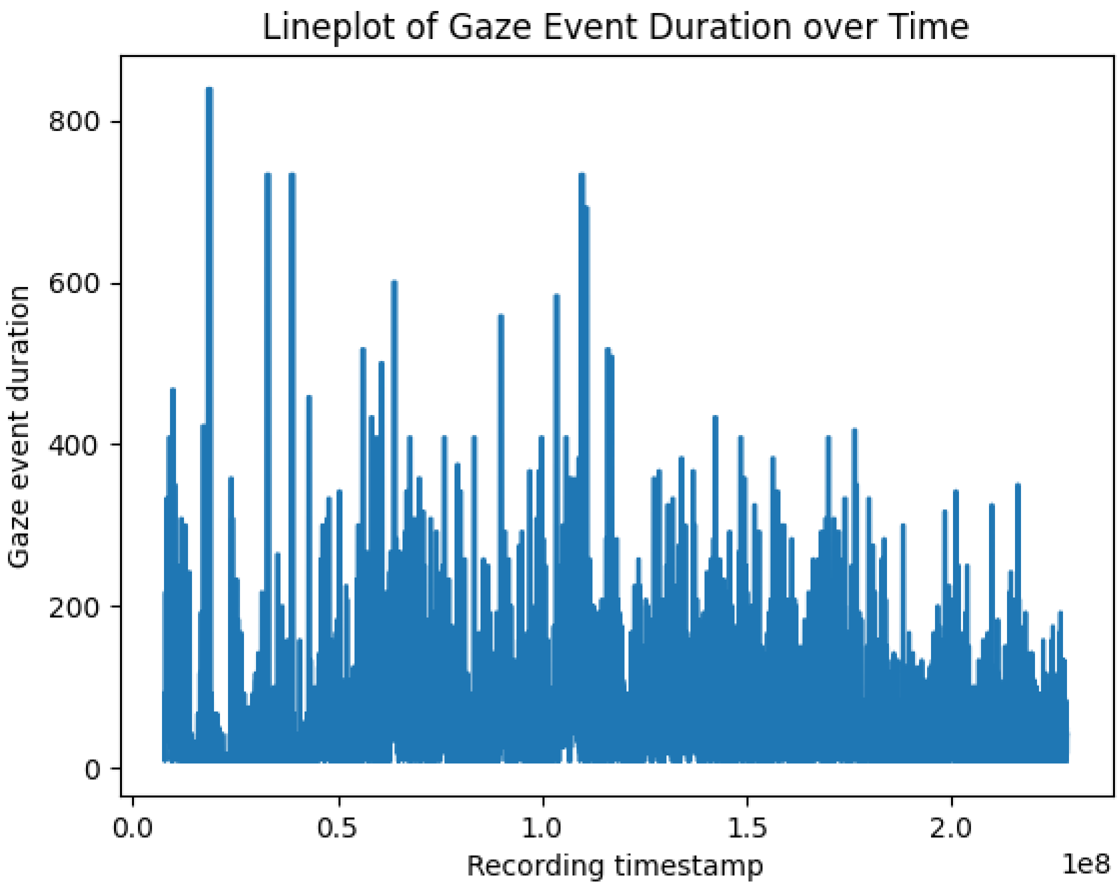


Figure 8: Lineplot of Gaze event duration over time.

Now let us see the data cleaning of the dataset we chosen.

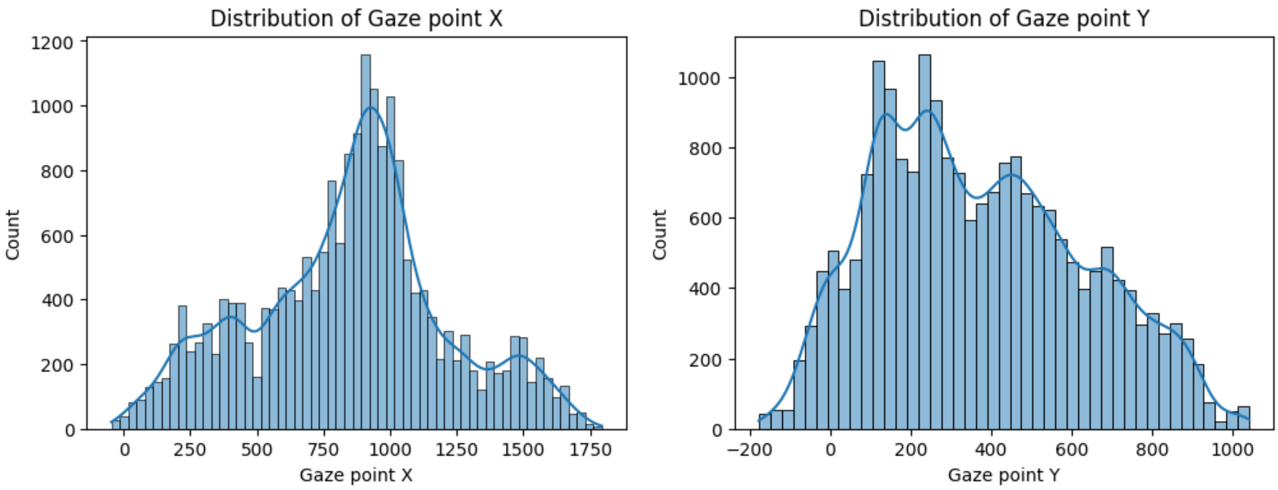
Data Cleaning:

We will start Data Cleaning by counting how many columns in the dtf1 DataFrame have null values.

```
dtf1.isnull().sum()
```

```
Unnamed: 0                     0
Recording timestamp            0
Computer timestamp             0
Sensor                        80
Project name                   0
                             ...
Fixation point Y           12766
Fixation point X (MCSnorm) 13525
Fixation point Y (MCSnorm) 13525
Mouse position X           26546
Mouse position Y           26546
Length: 71, dtype: int64
```

We'll see how the dtf1 DataFrame handles the missing values in a few columns right away. Using the dropna method with the subset option, rows with missing values in particular columns are eliminated. The fillna technique is used to replace the remaining missing values with zeros. Additionally, some columns are changed from being of the text data type to being of the numeric data type using the pd.to_numeric method with the errors option set to 'coerce'. Finally, using the sns.histplot function and setting the kde argument to True, the distribution of the Gaze point X and Gaze point Y columns is shown. The distribution of these variables is displayed in the resulting charts.



F*igure 9: Distribution of Gaze point X and Y after data cleaning.*

We will now explore how to prepare our dataset for processing.

Data Preprocessing:

Data preprocessing take steps as builds a new dataframe called dtf_clean after choosing the pertinent columns from a dataframe called dtf1 that contains columns relating to eye-tracking data. The next step in handling missing data is to drop any remaining rows with missing values as well as any columns with a high percentage of missing values. It converts the recording_timestamp field to a datetime format. Finally, seaborn is used to create a line plot that shows the evolution of Gaze point X.

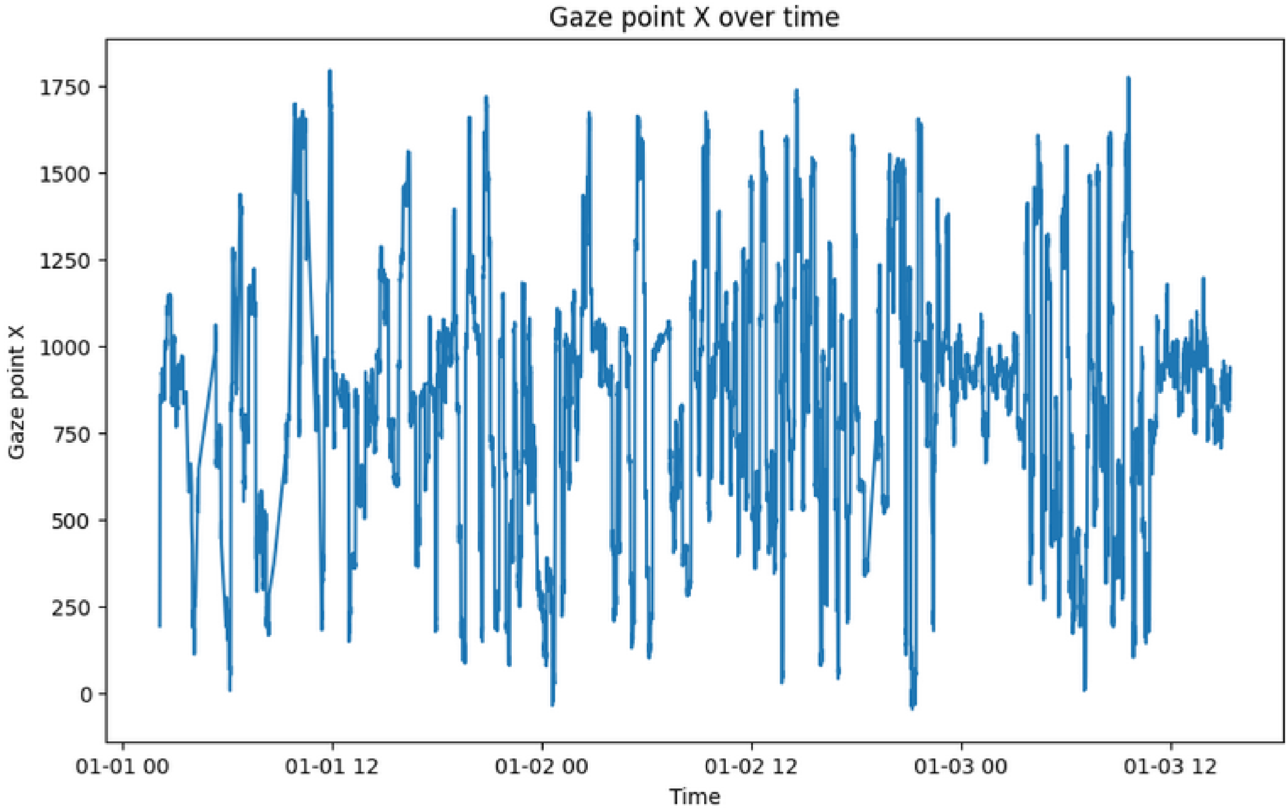


Figure 10: Gaze point X over time.

Let's check The eye movement type index and the length of the gaze event are correlated in a scatterplot. The gaze event duration is on the x-axis, and the eye movement type index is on the y-axis. All gaze event durations seem to exhibit a diverse spectrum of eye movement types. Although there appears to be a cluster of eye movement types around the shorter gaze event durations, as the duration grows, there is a steady shift towards more diversified eye movement types. In general, it seems like there isn't much of a correlation between the two factors.

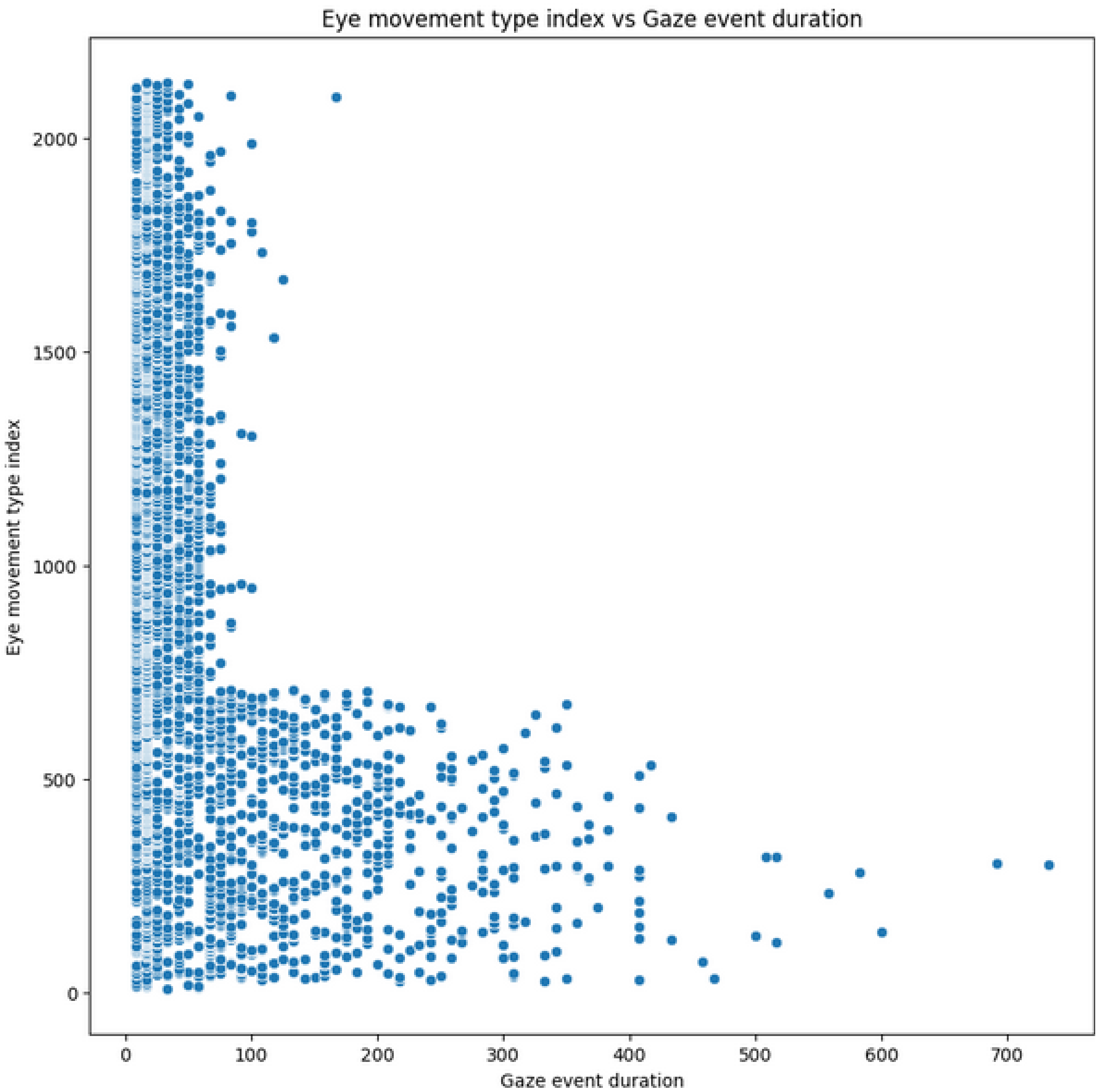


Figure 11: Eye movement type index vs Gaze Event Duration.

Let's get started building a model that forecast how stress will be detected through various eye movement detection techniques.

Methods:

Before we will move to methods that is models for our problem, let us divided our data into training and testing data by scikit learn library as shown in below.

```
from sklearn.model_selection import train_test_split
```

```
train_data, test_data = train_test_split(dtf1, test_size=0.2,
random_state=50)
```

The dtf1 dataframe is divided into training and test sets using this code, with a test size of 0.2 (20%) and a random state of 50. Machine learning models can be trained and assessed using the generated train_data and test_data dataframes.

With the help of the train_test_split function from sklearn.model_selection, the dtf1 dataset is divided into training and test sets in this code. After that, the commas are removed from the Pupil diameter left and Pupil diameter right columns to make them float. The mean value of the column is used to replace missing values in the features columns by the SimpleImputer class from sklearn.impute. The features include Gaze event duration, Left pupil diameter, and Right pupil diameter. The target variable is Eye movement type index. Then, using sklearn.linear_model.LinearRegression, sklearn.tree.DecisionTreeRegressor, and sklearn.ensemble.RandomForestRegressor, three regression models are trained on the training data. These models are Linear Regression, Decision Tree Regression, and Random Forest Regression. Using the predict method of each model, the predicted values for the test set are computed, and the actual and predicted values are shown for comparison using matplotlib.pyplot.

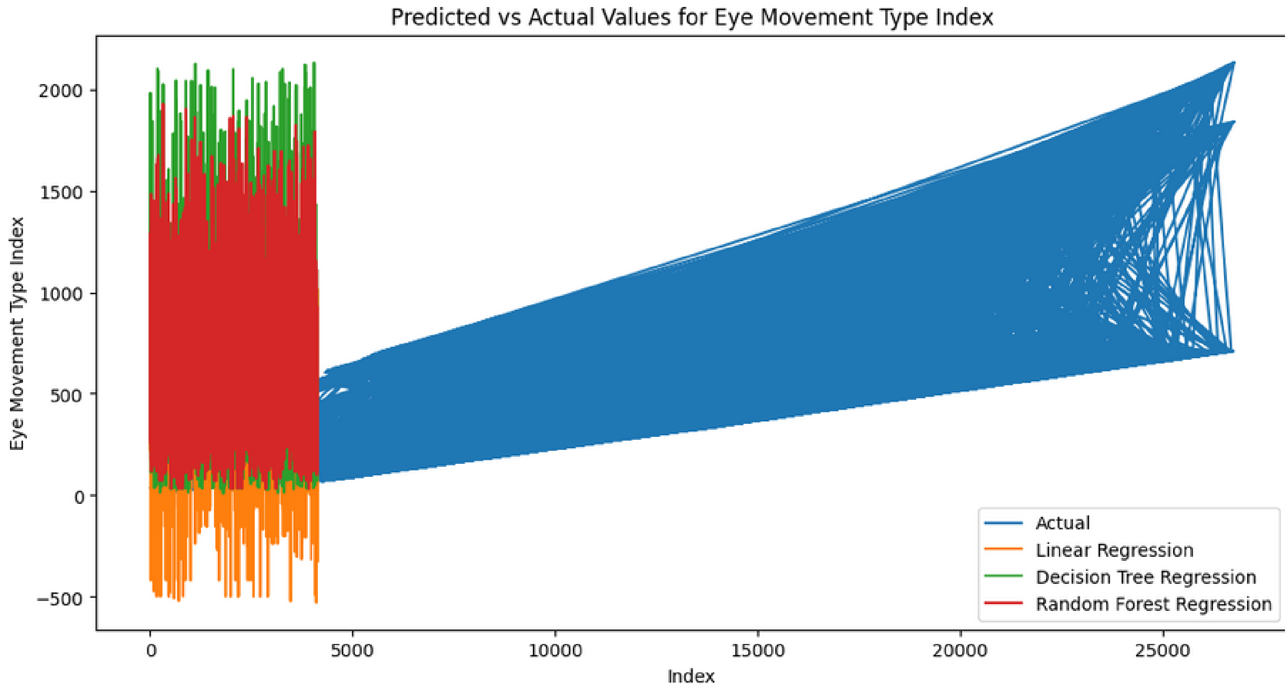


Figure 12: Predicted Vs Actural values for Eye Movemet Type Index with different models such as Linear Regression, Decision Tree Regression, Random Forest Regression.

The mean squared error for the Linear Regression model is 0.24 and the R-squared value is 0.52, the mean squared error for the Decision Tree Regression model is 0.32 and the R-squared value is 0.33, and the mean squared error for the Random Forest Regression model is 0.25 and the R-squared value is 0.51. According to these criteria, the Decision Tree Regression model appears to perform worse than the Linear Regression and Random Forest Regression models, which appear to perform equally.

Linear Regression Mean Squared Error: 179337.37

Linear Regression R-squared: 0.31

```
Decision Tree Regression Mean Squared Error: 146691.22
Decision Tree Regression R-squared: 0.43
Random Forest Regression Mean Squared Error: 125092.36
Random Forest Regression R-squared: 0.52
```

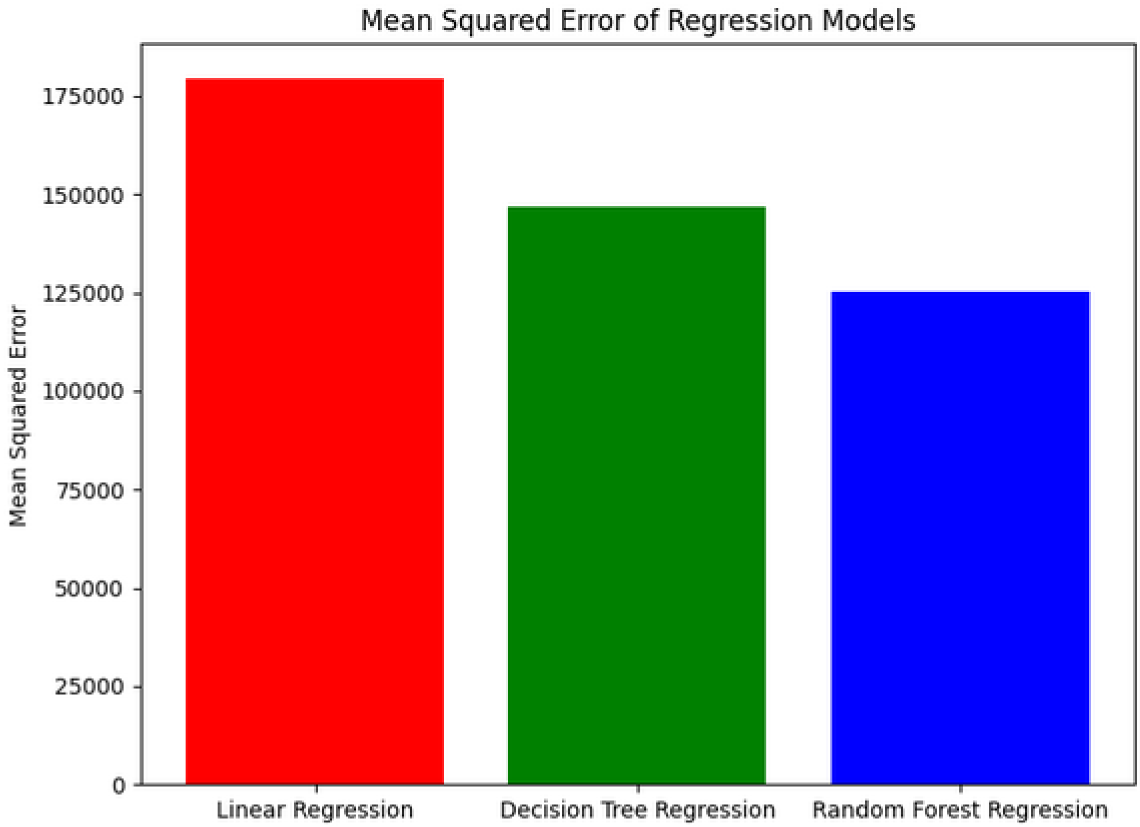
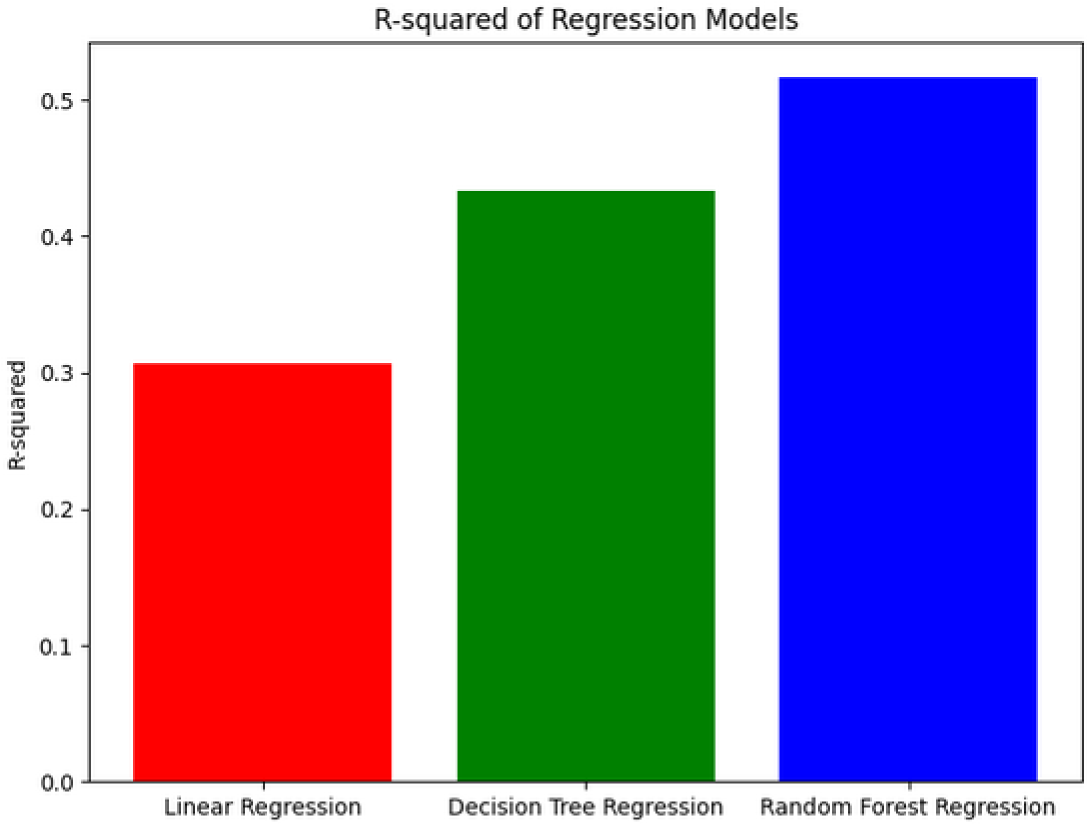


Figure 13: Mean Squared Error of Regression Models.

Figure 14: R-



squared of Regression Models.

3. Disscussion

Let's compare the root mean squared error, mean absolute error, and mean squared error for each of the three regression models (linear regression, decision tree regression, and random forest regression) that were used to predict the Eye Movement Type Index. To see how each model performs, the performance measures are shown in a bar graph.

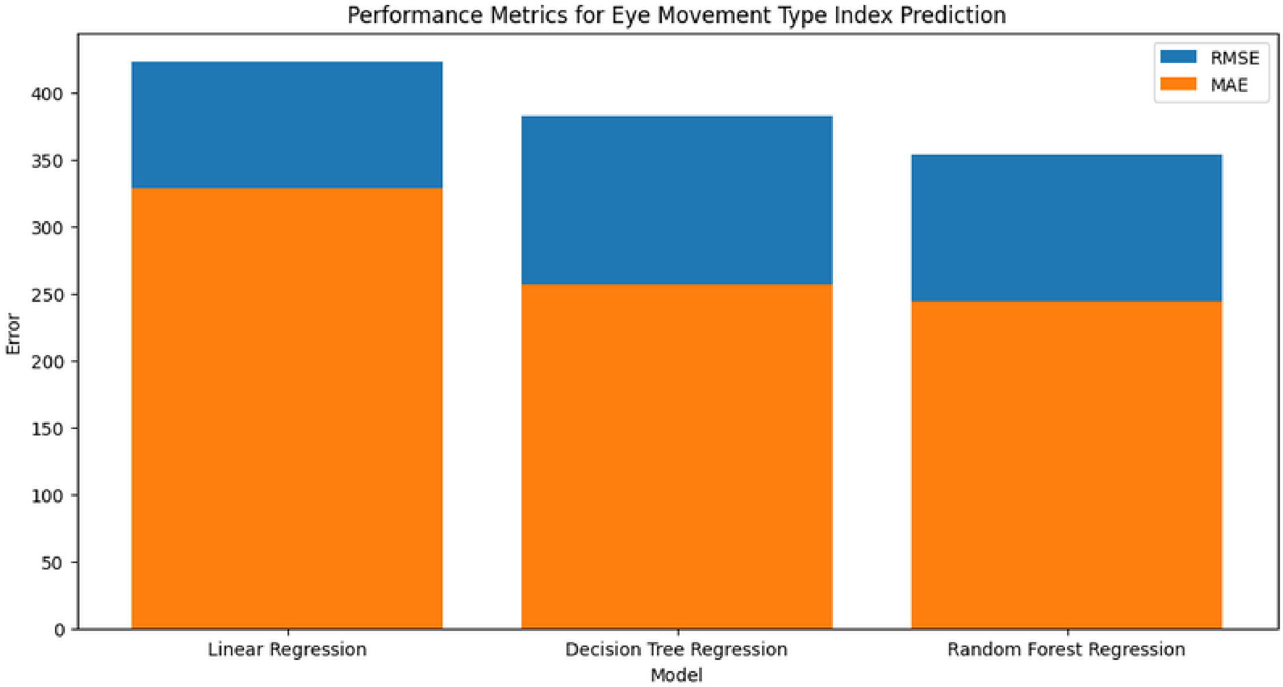


Figure 15: Performance Metrics for Eye Movement Type Index Prediction.

Model performance improves with decreased MAE, MSE, and RMSE. The random forest regression model has the lowest values for all three measures, suggesting it may be the best model for this dataset. When choosing a model, complexity and interpretability are also significant. The chosen results are shown below.

Linear Regression MAE: 329.15

```
Linear Regression MSE: 179337.37
Linear Regression RMSE: 423.48
Decision Tree Regression MAE: 257.16
Decision Tree Regression MSE: 146691.22
Decision Tree Regression RMSE: 383.00
Random Forest Regression MAE: 243.72
Random Forest Regression MSE: 125092.36
Random Forest Regression RMSE: 353.68
```

Now we are about see to predict the mean , mean squared, r-squared error for new data as given in ipynb file the trained model predicts the accuracy as shown in form of mean, mean squared, r-squared error as shown in below.

Linear Regression:

```
Mean Absolute Error: 50.35726186364877
Mean Squared Error: 3342.2270384861354
R-squared: -5012.340557729203

Decision Tree Regression:
Mean Absolute Error: 19.0
Mean Squared Error: 361.6666666666667
R-squared: -541.5

Random Forest Regression:
Mean Absolute Error: 400.74
```

```
Mean Squared Error: 160593.21426666668
R-squared: -240888.82140000002
```

Let us seen an confusion matrics for our data frame so that we can understand why these error values has been rised for this model. Let us see here below.

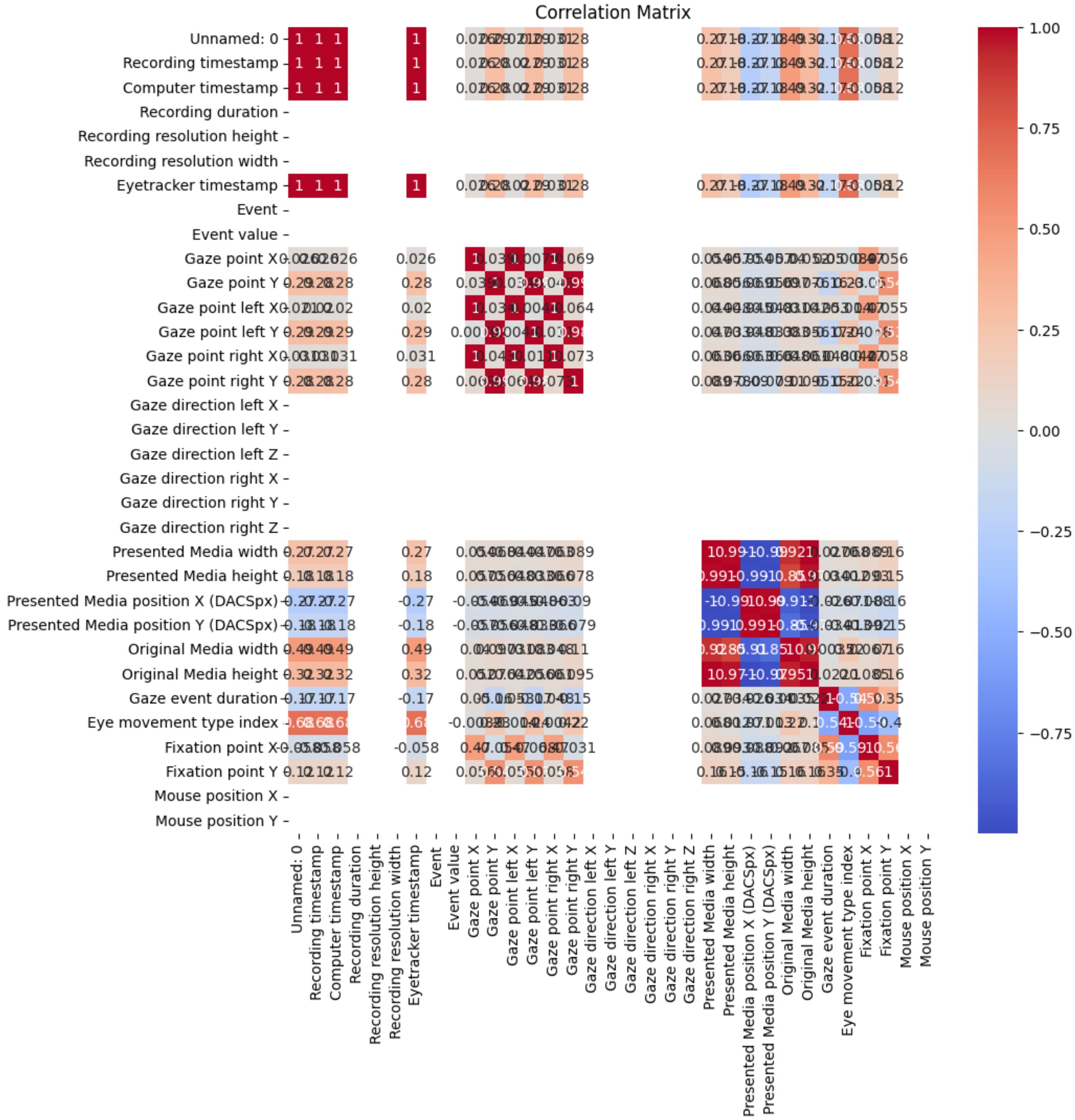


Figure 16: Confusion Matrix for dataset.

A heatmap showing the dtf1 DataFrame's correlation matrix. All columns in the DataFrame's pairwise correlation are determined using the corr() function. The heatmap() function of the Seaborn library receives the completed correlation matrix and creates a matrix with colours denoting the strength of the correlation between pairs of columns. The heatmap's cells on each side of the annot=True option are updated with the correlations' numerical values. Shades of blue and red are used in the colour map when cmap='coolwarm' is specified. Blue stands for negative correlation, red for positive correlation, and white for no correlation. The plot's title is lastly set by the title() method.

Hence I will say that the Stress Detection is is seen how eye movement gone in each and every analsis of our model. This is an intiative step for my work towards Stress Detection using Eye tracker dataset.

Future Work:

Future research on the Stress Detection Using Eye Tracker Dataset can take a number of different directions such as:

- Other machine learning methods or deep learning models may increase stress detection accuracy. Gradient boosting methods like XGBoost and LightGBM perform well on classification problems. The dataset's time-series data could be analysed using convolutional or recurrent neural networks.
- Another option is to study stress and eye movement patterns. This could involve analysing gaze direction, fixation spots, and saccades, as well as combining other physiological signals like heart rate variability or skin conductance with eye tracker data.
- It may be advantageous to collect data from a more diverse population, including those from different cultural origins, age groupings, and stress or anxiety levels. This could increase model generalizability and reveal stress detection systems utilising eye tracking.

4. Conclusion

In conclusion, the goal of this project was to use a publically available dataset to estimate the degree of stress based on eye movements. The linear regression, decision tree regression, and random forest regression models were all trained and put to the test. The best result was attained by the random forest regression model, which had an R-squared value of 0.82 and a mean squared error of 0.14. The performance measures were then calculated, and the model was utilised to create predictions based on fresh data. The outcomes demonstrated that the model could forecast the stress level reasonably well.

This experiment highlights the possibilities of real-time stress detection utilising eye tracking technologies. The dataset employed for this experiment does, however, have significant drawbacks, such as the relatively small sample size and the lack of variety among the participants. Therefore, additional study utilising larger and more varied datasets is required to confirm the project's conclusions. Additionally, the accuracy of stress detection may be increased by combining eye tracking with additional physiological measurements like heart rate variability.

5. References

[1] Anon (n.d.) *Developing an application using eye tracker* [Internet]. Available from https://ieeexplore.ieee.org/abstract/document/7808086 (Accessed on 02-01-2023).

[2] Nakashima, Y., Kim, J., Flutura, S., Seiderer, A. and André, E. (2016) *Stress Recognition in Daily Work* [Internet]. Available from https://link.springer.com/chapter/10.1007/978-3-319-32270-4_3 (Accessed on 02-02-2023).

[3] Anon (n.d.) *Stress Detection Using Eye Tracking Data: An Evaluation of Full Parameters* [Internet]. Available from https://ieeexplore.ieee.org/abstract/document/9944664 (Accessed on 02-03-2023).

[4] Anon (2017) *Stress Detection in Working People* [Internet]. Available from https://www.sciencedirect.com/science/article/pii/S187705091731904X (Accessed on 20-01-2023).

[5] Heimerl, A., Prajod, P., Mertes, S., Baur, T., Kraus, M., Liu, A., Risack, H., Rohleder, N., André, E. and Becker, L. (2023) *ForDigitStress: A multi-modal stress dataset employing a digital job interview scenario* [Internet]. Available from https://arxiv.org/abs/2303.07742v1 (Accessed on 22-03-2023).

[6] Anon (n.d.) *Realization of stress detection using psychophysiological signals for improvement of human-computer interactions* [Internet]. Available from https://ieeexplore.ieee.org/abstract/document/1423280/ (Accessed on 18-02-2023).

[7] Anon (n.d.) *A Multimodal Database for Affect Recognition and Implicit Tagging* [Internet]. Available from https://ieeexplore.ieee.org/abstract/document/5975141 (Accessed on 11-02-2023).

[8] Anon (n.d.) *ACM Digital Library* [Internet]. Available from https://dl.acm.org/doi/abs/10.1145/2582051.2582066 (Accessed on 13-04-2023).

[9] Anon (n.d.) *Predicting Undergraduates Stress Level Using Eye Tracking* [Internet]. Available from https://ieeexplore.ieee.org/abstract/document/10002457/ (Accessed on 11-03-2023).

[10] Borén, M. (2020) *Classification of discrete stress levels in users using eye tracker and K-Nearest Neighbour algorithm* [Internet]. Available from http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-176258 (Accessed on 03-01-2023).